ELabELN

Software License, Technical Specifications & Installation Manual

Ubuntu Linux • Docker Deployment

Complete Stack Inventory, Dependency Licenses & Step-by-Step Installation

Powered by LabLynx, Inc.

www.lablynx.com | www.elabeln.com | support@lablynx.com

Version 1.0 | June 2026 | CONFIDENTIAL

Table of Contents

Table of Contents 2
Part 1: Software License Information 3
1. ELabELN Commercial License (LabLynx, Inc.) 4
2. Open-Source Component Licenses 5
Part 2: Technical Specifications 15
3. System Architecture 16
4. Hardware and Software Requirements 18
Part 3: Installation Manual 21
5. Pre-Installation: Ubuntu Server Setup 22
6. Installing Docker Engine and Docker Compose 24
7. ELabELN Installation 27
8. Post-Installation Configuration 32
9. Updating ELabELN 35
10. Security Hardening 37
11. Monitoring and Health Checks 40
12. Troubleshooting 42

Part 1: Software License Information

This section describes the licensing terms governing ELabELN as delivered by LabLynx, Inc., and the licenses applicable to all open-source components included in the ELabELN software stack.

1. ELabELN Commercial License (LabLynx, Inc.)

1.1 License Summary

ELabELN is a commercial software product delivered by LabLynx, Inc. under a subscription license agreement. The commercial license governs the use of:

- LabLynx branding, customization, and configuration applied to the ELabELN platform.
- LabLynx-authored documentation, training materials, and support resources.
- LabLynx-managed hosting infrastructure (sciCloud.net®) where applicable.
- LabLynx proprietary integrations, connectors, and extensions.

ELabELN Commercial License — Key Terms

Licensor: LabLynx, Inc., Pensacola, Florida, USA

License Type: Commercial Subscription License

Grant: Non-exclusive, non-transferable right to use ELabELN as delivered by LabLynx for the subscriber's internal laboratory operations during the subscription term.

Users: Unlimited users per instance. No per-seat licensing.

Restrictions: Licensee may not redistribute, sublicense, sell, or host ELabELN as a service for third parties. Reverse engineering of LabLynx proprietary components is prohibited.

Support: Included per subscription tier. See www.lablynx.com for current tier details.

Contact: sales@lablynx.com | support@lablynx.com

1.2 Open-Source Foundation

ELabELN is built on eLabFTW, an open-source Electronic Laboratory Notebook developed by Deltablot SAS. LabLynx applies branding, configuration, enterprise features, and managed hosting to the eLabFTW platform. The underlying eLabFTW codebase is licensed under the GNU Affero General Public License v3.0 (AGPL-3.0).

eLabFTW Open Source License

Project: eLabFTW

Developer: Deltablot SAS

License: GNU Affero General Public License v3.0 (AGPL-3.0)

Source Code: https://github.com/elabftw/elabftw

AGPL-3.0 Summary: Permissions: Commercial use, distribution, modification, patent use, private use. Conditions: Disclose source, license and copyright notice, same license, state changes, network use is distribution. Limitations: No liability, no warranty.

Full License Text: https://www.gnu.org/licenses/agpl-3.0.html

1.3 License Compatibility Statement

LabLynx's commercial subscription does not alter the open-source licensing terms of eLabFTW or any of its open-source dependencies. Customers who deploy ELabELN using the LabServer (on-premises) model receive the complete source code of eLabFTW as part of the container images, in accordance with the AGPL-3.0 requirements. The full list of open-source dependency licenses is documented in Chapter 3 of this manual.

2. Open-Source Component Licenses

The ELabELN software stack incorporates numerous open-source components. The following sections document each component, its version, source, and applicable license. This inventory covers the Docker image contents, runtime dependencies, and infrastructure components.

2.1 License Type Summary

License	Description
GNU Affero General Public License v3.0 (AGPL-3.0)	Copyleft license requiring that the source code be made available when the software is run over a network. Applies to eLabFTW core.
GNU General Public License v2.0 (GPL-2.0) / v3.0 (GPL-3.0)	Copyleft license. Used by Linux kernel components within the Docker host environment.
GNU Lesser General Public License v2.1 / v3.0 (LGPL)	Weaker copyleft, permitting linking from non-GPL software. Used by several libraries.
MIT License	Permissive license allowing use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies. Very widely used.
BSD 2-Clause / BSD 3-Clause License	Permissive licenses similar to MIT with attribution requirements.
Apache License 2.0	Permissive license with patent grant. Used by Docker Engine and several PHP libraries.
Mozilla Public License 2.0 (MPL-2.0)	Weak copyleft applying at the file level. Used by some cryptographic components.
Creative Commons (CC0, CC-BY, CC-BY-SA)	Used for documentation, fonts, and data content included in the platform.
ISC License	Functionally equivalent to MIT. Used by several Node.js packages.
OpenSSL License (dual OpenSSL/SSLeay)	Legacy dual-license used by the OpenSSL library for TLS/cryptographic operations.

2.2 Core Application Stack — Primary Docker Image (elabftw/elabimg)

The primary ELabELN Docker image (elabftw/elabimg:stable) is built on Alpine Linux and bundles the following components:

Component	Version	License	Source / Homepage
eLabFTW	5.3.x (stable)	AGPL-3.0	https://github.com/elabftw/elabftw
Alpine Linux	3.20.x	MIT (OS components are mixed; kernel is GPL-2.0)	https://alpinelinux.org
nginx	1.26.x	BSD 2-Clause	https://nginx.org

PHP	8.3.x	PHP License v3.01	https://www.php.net/license/3_01.txt
PHP-FPM	8.3.x (bundled with PHP)	PHP License v3.01	https://www.php.net
s6-overlay	3.x	ISC License	https://github.com/just-containers/s6-overlay
skalibs	2.x	ISC License	https://skarnet.org/software/skalibs
execline	2.x	ISC License	https://skarnet.org/software/execline
s6	2.x	ISC License	https://skarnet.org/software/s6
OpenSSL	3.3.x	Apache License 2.0	https://openssl.org
zlib	1.3.x	zlib License (MIT-like)	https://zlib.net
libc (musl)	1.2.x	MIT License	https://musl.libc.org
libcurl	8.x	MIT License (curl)	https://curl.se/docs/copyright.html
GnuPG (gpg)	2.4.x	GPL-3.0	https://gnupg.org
Ghostscript	10.x	AGPL-3.0	https://www.ghostscript.com
ImageMagick	7.x	ImageMagick License (MIT-variant)	https://imagemagick.org/script/license.php
FreeType	2.x	FTL (FreeType License, BSD-like)	https://freetype.org/license.html
libpng	1.6.x	libpng License (BSD-like, permissive)	https://libpng.org
libjpeg-turbo	3.x	IJG / BSD 3-Clause / zlib	https://libjpeg-turbo.org
libzip	1.x	BSD 3-Clause	https://libzip.org
libxml2	2.12.x	MIT License	https://gitlab.gnome.org/GNOME/libxml2
libxslt	1.1.x	MIT License	https://gitlab.gnome.org/GNOME/libxslt
ICU (International Components for Unicode)	74.x	Unicode License (permissive)	https://icu.unicode.org
brotli	1.1.x	MIT License	https://github.com/google/brotli
borgbackup (optional)	1.4.x	BSD 3-Clause	https://borgbackup.readthedocs.io

2.3 PHP Extensions (Bundled in Container)

PHP Extension	License	Role in ELabELN
php8-bcmath	Arbitrary-precision arithmetic. MIT/PHP License.	Required for cryptographic operations.
php8-ctype	Character type checking. PHP License v3.01.	Required for input validation.
php8-curl	cURL HTTP client bindings. MIT/PHP License.	Required for API calls, timestamping.
php8-dom	XML Document Object Model. PHP License v3.01.	Required for HTML/XML processing.
php8-exif	EXIF image metadata reader. PHP License v3.01.	Required for image handling.
php8-fileinfo	File type detection. PHP License v3.01.	Required for file upload validation.
php8-gd	Image processing (GD library). PHP License v3.01 + MIT.	Required for image manipulation, PDF generation.
php8-gettext	Internationalization/localization . PHP License v3.01.	Required for multi-language support.
php8-iconv	Character encoding conversion. PHP License v3.01.	Required for encoding handling.
php8-intl	ICU internationalization. PHP License v3.01 + Unicode.	Required for locale-aware operations.
php8-json	JSON encoding/decoding. PHP License v3.01.	Required for API and data serialization.
php8-ldap	LDAP authentication. PHP License v3.01 + OpenLDAP.	Required for LDAP/AD integration.
php8-mbstring	Multibyte string handling. PHP License v3.01.	Required for Unicode text processing.
php8-opcache	PHP opcode caching. PHP License v3.01.	Performance: bytecode caching.
php8-openssl	OpenSSL cryptographic functions. PHP License v3.01 + Apache 2.0.	Required for TLS, encryption, e-signatures.
php8-pdo	PHP Data Objects (database abstraction). PHP License v3.01.	Required for database connectivity.
php8-pdo_mysql	MySQL PDO driver. PHP License v3.01.	Required for MySQL database access.
php8-session	PHP session management. PHP License v3.01.	Required for user sessions.
php8-simplexml	SimpleXML parser. PHP License v3.01.	Required for XML processing.

php8-sodium	libsodium cryptographic library binding. PHP License v3.01 + ISC.	Required for Ed25519 signing, encrypted storage.
php8-tokenizer	PHP tokenizer. PHP License v3.01.	Required internally.
php8-xml	XML processing. PHP License v3.01.	Required for XML output/processing.
php8-xmlwriter	XML writer. PHP License v3.01.	Required for XML generation.
php8-zip	ZIP archive handling. PHP License v3.01 + BSD 3-Clause.	Required for ZIP export/import.

2.4 PHP Composer Packages (Application-Level Dependencies)

The following PHP packages are installed via Composer during the Docker image build. All are governed by their respective open-source licenses.

Package	Version	License	Role in ELabELN
mpdf/mpdf	v8.2.x	GPL-2.0-only	PDF generation from HTML. Used for experiment PDF exports.
monolog/monolog	v3.x	MIT	Application logging framework.
symfony/console	v6.x / v7.x	MIT	CLI commands (database init, maintenance tools).
symfony/cache	v6.x / v7.x	MIT	Caching abstraction layer.
symfony/http-foundation	v6.x / v7.x	MIT	HTTP request/response abstraction.
symfony/mailer	v6.x / v7.x	MIT	Email sending (SMTP notifications, invitations).
symfony/mime	v6.x / v7.x	MIT	MIME type handling for email and file processing.
symfony/translation	v6.x / v7.x	MIT	Internationalization and translation framework.
symfony/yaml	v6.x / v7.x	MIT	YAML parsing.
twig/twig	v3.x	BSD 3-Clause	HTML template rendering engine.
league/commonmark	v2.x	BSD 3-Clause	Markdown to HTML conversion.
league/html-to-markdown	v5.x	MIT	HTML to Markdown conversion for exports.

intervention/image	v2.x / v3.x	MIT	Image manipulation and processing.
ramsey/uuid	v4.x	MIT	UUID generation for entry identifiers (ELabIDs).
paragonie/constant_time_encoding	v2.x	MIT	Constant-time string encoding for security.
paragonie/random_compat	v9.x	MIT	Cryptographically secure random number generation.
psr/log	v3.x	MIT	PSR-3 logging interface standard.
psr/cache	v3.x	MIT	PSR-6 cache interface standard.
psr/container	v2.x	MIT	PSR-11 container interface standard.
psr/http-message	v2.x	MIT	PSR-7 HTTP message interface standard.
php-http/httplug	v2.x	MIT	HTTP client abstraction.
guzzlehttp/guzzle	v7.x	MIT	HTTP client for external API calls (TSA, PubChem, etc.)
guzzlehttp/psr7	v2.x	MIT	PSR-7 HTTP message implementation.
php-di/php-di	v7.x	MIT	Dependency injection container.
defuse/php-encryption	v2.x	MIT	Symmetric AES-256 encryption for stored credentials.
minisign-php	v1.x	MIT	Ed25519 cryptographic signature verification.

2.5 JavaScript / Frontend Libraries (Bundled in Image)

Library	Version	License	Role in ELabELN
TinyMCE	v6.x / v7.x	MIT (Community)	Rich text WYSIWYG editor. Primary experiment body editor.
ChemDoodle Web Components	v9.x	GPL-3.0	3D molecular viewer for CIF/PDB/MOL files.
Ketcher (EPAM)	v2.x	Apache 2.0	2D chemical structure drawing editor.

3Dmol.js	v2.x	BSD 3-Clause	3D molecular visualization.
mermaid.js	v10.x	MIT	Diagram rendering from text markup.
SnapGene Viewer (embedded)	N/A	Proprietary free viewer	DNA sequence visualization (FASTA, .gb, .ape, .dna files).
Chart.js	v4.x	MIT	Canvas-based charting library.
XLSX.js (SheetJS)	v0.18.x	Apache 2.0 (Community)	Excel file parsing and export.
Bootstrap	v5.x	MIT	CSS framework for responsive UI components.
jQuery	v3.x	MIT	JavaScript utility library (legacy components).
Font Awesome	v6.x	MIT (icons), CC BY 4.0 (fonts)	Icon library used throughout the UI.
Mathjax	v3.x	Apache 2.0	LaTeX mathematical equation rendering.
CodeMirror	v6.x	MIT	Code block editor with syntax highlighting.
DOMPurify	v3.x	Apache 2.0 / MIT	HTML sanitization to prevent XSS attacks.
sortablejs	v1.x	MIT	Drag-and-drop sortable lists (steps, metadata groups).
interact.js	v1.x	MIT	Drag-and-resize interactions.
pako	v2.x	MIT	zlib/deflate compression in JavaScript.
lodash	v4.x	MIT	JavaScript utility library.
marked	v9.x	MIT	Markdown parser for preview mode.

2.6 Database Container (MySQL 8.4)

Component	Version	License
MySQL Community Server	8.4.x	GPL-2.0

mysql:8.4 Docker Image	8.4.x	GPL-2.0 (MySQL) + various OS licenses
MySQL Connector/C (libmysqlclient)	8.4.x	GPL-2.0 with FOSS Exception
OpenSSL (bundled in MySQL image)	3.x	Apache License 2.0

> **Note**
> MySQL 8.4 is the Long Term Support (LTS) release series. LabLynx recommends pinning to 8.4.x for production deployments. MySQL 8.0.x is still supported but will reach End of Life in April 2026. MySQL 9.x is the Innovation series and is not yet recommended for production ELabELN deployments.

2.7 Docker and Container Infrastructure

Component	Version	License	Role
Docker Engine	27.x	Apache License 2.0	Container runtime. Core of the deployment platform.
Docker Compose Plugin	2.x	Apache License 2.0	Multi-container orchestration. Manages web + mysql containers.
containerd	1.7.x	Apache License 2.0	Container runtime used by Docker Engine.
runc	1.1.x	Apache License 2.0	OCI runtime specification implementation. Runs containers.
libnetwork	0.8.x	Apache License 2.0	Docker networking layer.
Moby (Docker Engine source)	27.x	Apache License 2.0	Open-source Docker Engine codebase.
BuildKit	0.13.x	Apache License 2.0	Advanced build engine for Docker images (used at image build time).
CNI plugins	1.4.x	Apache License 2.0	Container Network Interface plugins.
iptables	1.8.x	GPL-2.0	Linux firewall/NAT rules for Docker networking.
Docker Hub Registry	N/A	Service terms / Apache 2.0 (client)	Container image registry hosting elabftw/elabimg and mysql images.

2.8 Ubuntu Host OS Components

Component	Version	License	Role
Ubuntu Server	24.04 LTS (Noble Numbat)	GPL (kernel), MIT, BSD, Apache 2.0 (userspace varies)	Host operating system. Long Term Support until April 2029.
Linux Kernel	6.8.x	GPL-2.0-only	OS kernel providing system calls, container namespaces, cgroups.
systemd	255.x	LGPL-2.1+	Service manager. Manages Docker daemon as a service.
OpenSSH Server	9.6.x	BSD-style (OpenSSH license)	Remote administration access. Separate from application TLS.
ufw (Uncomplicated Firewall)	0.36.x	GPL-3.0	Host-level firewall configuration utility.
AppArmor	4.0.x	GPL-2.0	Mandatory Access Control framework for container security profiles.
curl	8.5.x	MIT (curl License)	Used by elabctl installer to fetch configuration.
openssl	3.0.x	Apache License 2.0	Host TLS utilities (certificate management, key generation).
apt	2.7.x	GPL-2.0	Ubuntu package manager for installing host dependencies.
gpg / gnupg2	2.4.x	GPL-3.0	GPG key verification for Docker repository authentication.
ca-certificates	20240203	MPL-2.0 (Mozilla), Various	Trusted root certificate bundle for TLS verification.
borgbackup	2.0.x	BSD 3-Clause	Optional backup tool for ELabELN data.
dialog	1.3.x	LGPL-2.0+	TUI dialog boxes used by elabctl interactive installer.
cron	3.0pl1	ISC	Scheduled task execution for automated backups.
logrotate	3.21.x	GPL-2.0	Log file rotation for Docker and application logs.

fail2ban	1.0.x	GPL-2.0	Intrusion prevention: blocks IPs after repeated failed auth.

2.9 Optional Add-On Components

Component	Version	License	Role
Redis (redis:7-alpine image)	7.2.x	BSD 3-Clause	Optional session storage backend. Required for multi-container / high-availability deployments.
Indigo Chemistry Plugin (elabftw/chem-plugin)	latest	Apache License 2.0 (EPAM INDIGO)	Chemical structure editor (Ketcher) and substructure search (fingerprinting).
OpenCloning Plugin	latest	MIT	DNA cloning workflow integration.
Let's Encrypt (Certbot)	2.x	Apache License 2.0	Free TLS certificate issuance and auto-renewal. Required if ENABLE_LETSENCRYPT=true.
Nginx (reverse proxy, optional)	1.26.x	BSD 2-Clause	If using a reverse proxy in front of ELabELN (TLS termination, load balancing).
Traefik (optional reverse proxy)	3.x	MIT	Alternative reverse proxy with built-in Let's Encrypt support.
Certbot ACME client	2.x	Apache License 2.0	ACME protocol client for certificate management.

2.10 Compliance Note on AGPL-3.0

ELabELN includes eLabFTW software licensed under the GNU Affero General Public License version 3.0 (AGPL-3.0). The AGPL-3.0 requires that:

- If you run a modified version of eLabFTW over a network, you must make the source code available to users of that network service.
- LabLynx, as the distributor of an unmodified (branded) version of eLabFTW, satisfies this requirement by providing the source code via the public eLabFTW GitHub repository (https://github.com/elabftw/elabftw).
- Customers who deploy ELabELN on-premises (LabServer) must ensure that any modifications they make to the eLabFTW source code are made available under the same AGPL-3.0 terms if those modifications are used to provide network services.
- LabLynx proprietary components (branding, configuration files, integration scripts) that do not constitute derived works of eLabFTW are not subject to AGPL-3.0.

> **Note**
> The complete text of all open-source licenses is available in the /licenses directory of the eLabFTW source repository at https://github.com/elabftw/elabftw and in the /usr/share/doc directory of the deployed Docker container.

Part 2: Technical Specifications

3. System Architecture

3.1 Application Architecture Overview

ELabELN is a three-tier web application deployed using Docker containerization. The architecture consists of:

- Presentation Tier: Nginx web server serving static assets and proxying PHP requests.
- Application Tier: PHP 8.3 running via PHP-FPM (FastCGI Process Manager), executing the eLabFTW application logic.
- Data Tier: MySQL 8.4 database storing all application data, user records, experiments, resources, and audit logs.

All three components are orchestrated by Docker Compose on a single Ubuntu Linux host for standard deployments. The web/application tier runs in one Docker container (elabftw/elabimg); the database tier runs in a separate Docker container (mysql:8.4). Both containers communicate over an isolated internal Docker bridge network (elabftw-net).

Item	Description
Layer	Component

Layer	Component	Image	Port Exposure
Web + App	nginx + php-fpm + eLabFTW	elabftw/elabimg:stable	443 (HTTPS)
Database	MySQL 8.4	mysql:8.4	None (unexposed)
Session Store (optional)	Redis 7	redis:7-alpine	None (unexposed)
Chemistry Plugin (optional)	Indigo + Ketcher	elabftw/chem-plugin	None (unexposed)

Request Flow

1. User browser sends HTTPS request to server IP/domain on port 443.
2. Nginx receives the request. If it is a static file (CSS, JS, image), nginx serves it directly from the container filesystem. If it is a PHP request, nginx forwards it to PHP-FPM over the Unix socket.
3. PHP-FPM runs the eLabFTW PHP code, which processes the request, queries MySQL as needed, and generates an HTML response.
4. The response passes back through PHP-FPM to nginx, which returns it to the browser with appropriate headers.
5. All database reads/writes occur over the elabftw-net Docker bridge network between the web container and the mysql container.

3.2 Internal Container Services (s6-overlay)

The elabftw/elabimg container runs four concurrent services managed by s6-overlay (the container process supervisor):

Service	Description
nginx	Web server. Handles TLS, static file serving, and PHP-FPM proxying. Configuration in /etc/nginx/.
php-fpm	PHP FastCGI Process Manager. Executes the ELabELN PHP application. Manages a pool of PHP worker processes. Configuration in /etc/php83/php-fpm.d/.
cron (fcron)	Runs scheduled tasks: session garbage collection, temporary file cleanup, and any configured automation.
php-fpm status (optional)	Exposes a /php-status endpoint (password-protected) for PHP-FPM pool metrics monitoring.

3.3 Data Persistence Model

Two Docker bind mounts provide persistent storage outside the containers:

Mount	Description
File Uploads (/elabftw/uploads)	Container path mapped to /var/elabftw/web on the host. Stores all user-uploaded files, attachment exports, and generated ZIPs. This directory must be preserved across container updates.
MySQL Data (/var/lib/mysql)	Container path mapped to /var/elabftw/mysql on the host. Stores the complete MySQL database. This is the primary data directory. Must be backed up regularly.

> **Important**
> **Both bind mount directories (/var/elabftw/web and /var/elabftw/mysql) must exist on the host before starting containers, and must be included in all backup procedures. Data stored only inside containers is lost when containers are removed or recreated.**

3.4 Networking Architecture

Network Component	Description
elabftw-net	Internal Docker bridge network. Carries all communication between the web container and the mysql container. Not accessible from outside the host. The mysql container does NOT expose port 3306 externally — only the web container can connect to it.
Host Network (port 443)	The web container exposes port 443 (HTTPS) to the Docker host, which then routes it to incoming requests. In a security-hardened deployment, restrict this to a specific interface (127.0.0.1:443:443) and use a reverse proxy for external access.
Reverse Proxy (optional)	For deployments behind a reverse proxy (nginx, Traefik, HAProxy), set DISABLE_HTTPS=true and configure the proxy to handle TLS. Use ALLOW_ORIGIN, ALLOW_METHODS, and ALLOW_HEADERS environment variables for CORS if the API is accessed from a different domain.

4. Hardware and Software Requirements

4.1 Minimum Hardware Requirements

Component	Specification
CPU Architecture	64-bit x86_64 (AMD64). ARM64 is supported but not the primary tested architecture. 32-bit architectures are not supported.
CPU Cores	Minimum 2 cores. Recommended 4+ cores for production use. PHP-FPM and MySQL both benefit from additional cores.
RAM	Minimum 2 GB. Recommended 4 GB for a typical lab (up to ~50 concurrent users). 8 GB+ for larger deployments or high attachment volumes. The PHP_MAX_MEMORY setting defaults to 2 GB (the process memory limit, not total consumption).
Disk — OS + Docker	Minimum 20 GB for the operating system, Docker, and container images. Recommended 40+ GB.
Disk — Application Data	Varies by usage. Plan 1 GB per 100 experiments with typical attachments. For a 10-person lab with large instrument data files, plan 100 GB–1 TB minimum on the data partition (/var/elabftw). Use network storage (NFS, SAN) for large deployments.
Network	At minimum 100 Mbps LAN connection. Gigabit Ethernet recommended. Public-facing deployments require a static public IP or DNS-resolvable domain name.

> **Note**
> The PHP memory limit (MAX_PHP_MEMORY) and the number of PHP-FPM child processes (PHP_MAX_CHILDREN) are the primary performance tuning parameters. On a server with 4 GB RAM, a PHP_MAX_CHILDREN of 20 and MAX_PHP_MEMORY of 512M is a reasonable starting point. Monitor /php-status for pool utilization under load.

4.2 Recommended Production Configurations

Deployment Scale	Hardware Recommendation	Key Configuration Parameters
Small Lab (1–10 users)	4 CPU cores, 4 GB RAM, 100 GB SSD data disk. Ubuntu 24.04 LTS VPS or physical server.	PHP_MAX_CHILDREN=20, MAX_PHP_MEMORY=512M
Medium Lab (10–50 users)	8 CPU cores, 8 GB RAM, 500 GB SSD data disk (or network storage). Dedicated server or cloud VM.	PHP_MAX_CHILDREN=50, MAX_PHP_MEMORY=1G
Large Lab (50–200+ users)	16+ CPU cores, 16+ GB RAM, 1+ TB storage (NFS/SAN). Consider multiple containers with Redis session store.	PHP_MAX_CHILDREN=100, MAX_PHP_MEMORY=2G, USE_REDIS=true

High-Availability	2+ application servers with load balancer, shared NFS for uploads, external MySQL (MySQL Group Replication or PaaS). Redis required for session sharing.	USE_REDIS=true, load balancer with sticky sessions or Redis-based session store

4.3 Software Prerequisites (Host OS)

Requirement	Specification
Operating System	Ubuntu Server 24.04 LTS (Noble Numbat) — 64-bit x86_64. Fully patched with all security updates applied.
Docker Engine	Docker CE (Community Edition) 27.x or later. Installed via Docker's official apt repository. NOT the snap package.
Docker Compose Plugin	v2.x (Docker Compose V2). Installed as a Docker CLI plugin via the docker-compose-plugin apt package.
curl	8.5.x or later. For downloading configuration files and the elabctl installer.
dialog	1.3.x or later. For the interactive elabctl installation wizard.
bash	5.2.x or later. For running elabctl and installation scripts.
borgbackup (optional)	2.x or later. Required if using elabctl backup.
openssl	3.0.x or later. For TLS certificate management and key generation.
Domain Name	A fully qualified domain name (FQDN) resolving to the server's IP address is required for HTTPS. Required for Let's Encrypt certificate issuance.
Outbound Internet Access	Required for: Docker image pulls from Docker Hub, Let's Encrypt certificate validation, RFC 3161 timestamp requests, PubChem compound lookup, and software updates.

4.4 Network Ports and Firewall Requirements

Port	Protocol	Direction	Purpose
443 (TCP)	HTTPS	Inbound	Required. All user browser traffic. ELabELN web interface and REST API.
80 (TCP)	HTTP	Inbound	Optional. Required only if Let's Encrypt HTTP-01 validation is used. Redirect to 443.
22 (TCP)	SSH	Inbound	Recommended. Server administration. Restrict to admin IPs only.
3306 (TCP)	MySQL	Internal only	Must NOT be exposed externally. Only elabftw-net container traffic.
6379 (TCP)	Redis	Internal only	Must NOT be exposed externally. Only if Redis is deployed.

Outbound 443	HTTPS	Outbound	Required. Docker Hub image pulls, Let's Encrypt, TSA (RFC 3161), PubChem API.
Outbound 25/587	SMTP	Outbound	Required for email notifications if using external SMTP relay.

4.5 TLS / SSL Certificate Requirements

ELabELN requires HTTPS for all production deployments. Three TLS configuration options are supported:

TLS Option	Configuration and Requirements
Let's Encrypt (automated, recommended)	Set ENABLE_LETSENCRYPT=true. ELabELN handles certificate issuance and auto-renewal via the ACME protocol. Requires a public domain name, outbound port 443, and port 80 for initial HTTP-01 validation. Certificates are automatically renewed before expiry.
Custom Certificate (bring your own)	Place fullchain.pem and privkey.pem in /etc/letsencrypt/live/<SERVER_NAME>/ on the host. Mount /etc/letsencrypt:/ssl in the docker-compose.yml. Supports certificates from any CA (DigiCert, Sectigo, GlobalSign, internal CA, etc.).
No TLS (reverse proxy handles it)	Set DISABLE_HTTPS=true. nginx inside the container listens on 443 as HTTP (not HTTPS). The reverse proxy (nginx, Traefik, HAProxy) handles TLS termination and forwards plain HTTP to the container. Use this for enterprise environments with centralized certificate management.

Part 3: Installation Manual

This section provides complete, step-by-step installation instructions for ELabELN on Ubuntu Server 24.04 LTS using Docker. All commands are shown as they should be entered in a terminal. Commands prefixed with # are comments. Commands run as root are prefixed with #; commands run as a regular user are prefixed with $.

Before You Begin

Ensure you have the following ready before starting the installation:

- A 64-bit Ubuntu Server 24.04 LTS system with root (sudo) access.
- A fully qualified domain name (FQDN) pointing to your server's public IP.
- Outbound internet connectivity on port 443 (HTTPS).
- At least 4 GB RAM, 4 CPU cores, and 100 GB available disk space.
- This manual assumes Ubuntu Server 24.04 LTS with a fresh installation. Estimated installation time: 30–60 minutes.

5. Pre-Installation: Ubuntu Server Setup

5.1 Update the Operating System

Always start with a fully updated system. Log in to your Ubuntu server as a user with sudo privileges and run:

```
# Update the package index and upgrade all installed packages
sudo apt-get update
sudo apt-get upgrade -y

# Install a reboot if the kernel was updated
sudo reboot

# After reboot, verify the Ubuntu version
lsb_release -a
# Expected output: Ubuntu 24.04.x LTS (Noble Numbat)

# Verify architecture (must be x86_64 / amd64)
uname -m
# Expected: x86_64
```

5.2 Configure the Hostname and DNS

Set the server's hostname to match your planned domain name. This is used by Let's Encrypt and nginx configuration.

```
# Set the hostname (replace elab.yourlab.com with your actual FQDN)
sudo hostnamectl set-hostname elab.yourlab.com

# Verify
hostname -f
# Expected: elab.yourlab.com

# Add to /etc/hosts (recommended for local resolution fallback)
echo '127.0.1.1  elab.yourlab.com' | sudo tee -a /etc/hosts

# Verify your domain resolves to this server's public IP from external DNS
# (Run this from your local machine or any external host, not the server itself)
# nslookup elab.yourlab.com
# Expected: returns your server's public IP
```

5.3 Configure the Firewall (ufw)

Ubuntu 24.04 includes ufw (Uncomplicated Firewall). Configure it to allow only the necessary ports before installing Docker (Docker will add its own iptables rules separately):

```
# Enable ufw
sudo ufw enable

# Allow SSH (always allow SSH before enabling ufw to avoid locking yourself out)
sudo ufw allow 22/tcp
```

```
# Allow HTTPS (required)
sudo ufw allow 443/tcp

# Allow HTTP (required for Let's Encrypt HTTP-01 challenge)
# Only needed if using Let's Encrypt; can be removed after certificate issuance
sudo ufw allow 80/tcp

# Reload ufw
sudo ufw reload

# Verify status
sudo ufw status verbose
# Expected output shows rules for 22, 80, 443
```

Important

Docker manages its own iptables rules and adds them directly without going through ufw. This means Docker-exposed ports bypass ufw. To prevent this, either: (1) restrict Docker port bindings to 127.0.0.1 (e.g., '127.0.0.1:443:443') and use a reverse proxy, or (2) install the ufw-docker package. This is documented in Section 10.3 (Security Hardening).

5.4 Create the ELabELN Data Directories

Create the host directories that will be bind-mounted into the Docker containers for persistent storage:

```
# Create the parent data directory
sudo mkdir -p /var/elabftw

# Create subdirectories for uploads (application files) and MySQL data
sudo mkdir -p /var/elabftw/web
sudo mkdir -p /var/elabftw/mysql

# Set permissions: the elabftw container runs as the 'nginx' user (UID 101)
# The uploads directory needs to be owned by UID 101
sudo chown -R 101:101 /var/elabftw/web

# MySQL manages its own directory ownership inside the container
# No special ownership change is needed for the mysql directory

# Verify
ls -la /var/elabftw/
# Expected:
# drwxr-xr-x  nginx nginx  /var/elabftw/web
# drwxr-xr-x  root  root   /var/elabftw/mysql
```

Note

If your uploads directory is on an NFS mount point with enforced UIDs, adjust ELABFTW_USER, ELABFTW_GROUP, ELABFTW_USERID, and ELABFTW_GROUPID environment variables in the docker-compose.yml to match the NFS-imposed ownership.

6. Installing Docker Engine and Docker Compose

> **Important**
> **Do NOT install Docker via snap (sudo snap install docker). The snap-packaged Docker is known to cause permission and volume mounting issues with ELabELN. Always install Docker from the official Docker apt repository as described below.**

6.1 Remove Any Conflicting Packages

Ubuntu 24.04 may have older or conflicting Docker packages from the default apt repository. Remove them before installing from the Docker repository:

```
# Remove any conflicting Docker-related packages
for pkg in docker.io docker-doc docker-compose docker-compose-v2 \
           podman-docker containerd runc; do
  sudo apt-get remove -y $pkg 2>/dev/null || true
done

# Confirm removal
docker --version 2>&1 || echo 'Docker not installed (expected)'
```

6.2 Add the Docker Official APT Repository

```
# Step 1: Install prerequisite packages
sudo apt-get install -y \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

# Step 2: Create the Docker keyring directory
sudo install -m 0755 -d /etc/apt/keyrings

# Step 3: Download and store Docker's official GPG signing key
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
     -o /etc/apt/keyrings/docker.asc

# Step 4: Set correct permissions on the key file
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Step 5: Add the Docker stable repository to apt sources
echo \
  "deb [arch=$(dpkg --print-architecture) \
  signed-by=/etc/apt/keyrings/docker.asc] \
  https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Step 6: Update the package index
sudo apt-get update
```

6.3 Install Docker Engine and Compose Plugin

```
# Install the latest stable Docker Engine and Compose plugin
sudo apt-get install -y \
    docker-ce \
    docker-ce-cli \
    containerd.io \
    docker-buildx-plugin \
    docker-compose-plugin

# Verify Docker Engine is installed and running
sudo docker version
# Expected: Client and Server version information showing Docker CE 27.x or later

# Verify Docker Compose plugin (V2)
docker compose version
# Expected: Docker Compose version v2.x.x

# Verify Docker service is enabled and running
sudo systemctl status docker
# Expected: Active: active (running)
```

6.4 Configure Docker to Start on Boot

```
# Enable Docker to start automatically on system boot
sudo systemctl enable docker
sudo systemctl enable containerd

# Verify
sudo systemctl is-enabled docker
# Expected: enabled
```

6.5 (Optional) Allow Non-Root Docker Access

To run Docker commands without sudo for the administrative user, add the user to the docker group:

```
# Add your user to the docker group (replace 'ubuntu' with your username)
sudo usermod -aG docker ubuntu

# Apply group membership (log out and back in, or use newgrp)
newgrp docker

# Test non-root Docker access
docker run --rm hello-world
# Expected: 'Hello from Docker!' message
```

> **Caution**
> Adding a user to the docker group is effectively equivalent to giving them root access to the host system (Docker can mount host filesystems and break out of container isolation without additional security measures). Only add trusted administrators to the docker group in production environments.

6.6 Install elabctl (ELabFTW Control Tool)

elabctl is LabLynx's recommended command-line management tool for ELabELN (based on the eLabFTW elabctl script). It handles initial installation, updates, backups, and maintenance tasks.

```
# Install dialog (required by elabctl interactive installer)
sudo apt-get install -y dialog

# Download elabctl
curl -sO https://get.elabftw.net/elabctl.sh

# Review the script before running it (best practice)
cat elabctl.sh | less

# Install elabctl to /usr/bin/elabctl
sudo install -m 755 elabctl.sh /usr/bin/elabctl

# Verify installation
elabctl --version
# Expected: version number output

# View available commands
elabctl help
```

7. ELabELN Installation

7.1 Download the Docker Compose Configuration

Download the official ELabELN Docker Compose configuration file. This file defines all containers, environment variables, volumes, and network configuration:

```
# Create the ELabELN configuration directory
sudo mkdir -p /etc/elabftw

# Download the docker-compose.yml configuration file
# This fetches the template from LabLynx/eLabFTW with a randomly generated
# SECRET_KEY and DB_PASSWORD pre-populated
sudo curl -so /etc/elabftw/docker-compose.yml \
     "https://get.elabftw.net/?config"

# Verify the file was downloaded
sudo cat /etc/elabftw/docker-compose.yml | head -20
```

7.2 Configure docker-compose.yml

Edit the docker-compose.yml file to configure ELabELN for your environment. The minimum required settings are:

```
# Open the configuration file for editing
sudo nano /etc/elabftw/docker-compose.yml

# --- MINIMUM REQUIRED CHANGES ---

# 1. Under the 'web' service, environment section:
#    Set DB_PASSWORD to a strong random password
#    (a randomly generated one may already be present from the download)
- DB_PASSWORD=CHANGE_THIS_TO_A_STRONG_PASSWORD

# 2. Set SITE_URL to your server's FQDN (including https://)
- SITE_URL=https://elab.yourlab.com

# 3. Set PHP_TIMEZONE and TZ to your local timezone
#    Full list: http://php.net/manual/en/timezones.php
- PHP_TIMEZONE=America/Chicago
- TZ=America/Chicago

# 4. Set SERVER_NAME to your FQDN (without https://)
- SERVER_NAME=elab.yourlab.com

# 5. For Let's Encrypt TLS (recommended for public servers):
- ENABLE_LETSENCRYPT=true

# 6. Under the 'mysql' service, environment section:
#    Set MYSQL_ROOT_PASSWORD to a strong random password
- MYSQL_ROOT_PASSWORD=CHANGE_THIS_TO_ANOTHER_STRONG_PASSWORD
#    Set MYSQL_PASSWORD to MATCH the DB_PASSWORD from step 1
- MYSQL_PASSWORD=SAME_AS_DB_PASSWORD_ABOVE
#    Set TZ to match
- TZ=America/Chicago
```

> **Important**
> **The DB_PASSWORD value in the 'web' service environment MUST match the MYSQL_PASSWORD value in the 'mysql' service environment. If these differ, the web container cannot connect to the database and ELabELN will not start.**

If using a custom TLS certificate (not Let's Encrypt), add the SSL volume mapping under the 'web' service:

```
# Under 'web' service, volumes section, add:
    volumes:
        - /var/elabftw/web:/elabftw/uploads
        - /etc/letsencrypt:/ssl   # <-- add this line

# Ensure your certificate files exist at:
# /etc/letsencrypt/live/elab.yourlab.com/fullchain.pem
# /etc/letsencrypt/live/elab.yourlab.com/privkey.pem

# The SERVER_NAME value must match the certificate's Common Name
```

7.3 Complete Annotated docker-compose.yml

The following is the production-ready docker-compose.yml for a typical ELabELN deployment with Let's Encrypt TLS:

```
# /etc/elabftw/docker-compose.yml
# ELabELN Production Configuration
# LabLynx, Inc. — June 2026

name: elabftw

networks:
  elabftw-net:

services:
  web:
    image: elabftw/elabimg:stable
    restart: always
    container_name: elabftw
    depends_on:
      mysql:
        condition: service_healthy
    security_opt:
      - no-new-privileges:true
    cap_drop:
      - ALL
    cap_add:
      - CHOWN
      - SETGID
      - SETUID
      - FOWNER
      - DAC_OVERRIDE
    environment:
      # ---- MySQL Connection ----
      - DB_HOST=mysql
      - DB_PORT=3306
      - DB_NAME=elabftw
      - DB_USER=elabftw
      - DB_PASSWORD=CHANGE_THIS_STRONG_PASSWORD
```

```
      # ---- PHP ----
      - PHP_TIMEZONE=America/Chicago
      - TZ=America/Chicago
      - PHP_MAX_CHILDREN=50
      - PHP_MAX_EXECUTION_TIME=120
      - MAX_PHP_MEMORY=1G
      - MAX_UPLOAD_SIZE=100M
      # ---- ELabELN ----
      - SECRET_KEY=GENERATED_DURING_DOWNLOAD
      - SITE_URL=https://elab.yourlab.com
      # ---- nginx ----
      - SERVER_NAME=elab.yourlab.com
      - DISABLE_HTTPS=false
      - ENABLE_LETSENCRYPT=true
      # ---- Auto-maintenance ----
      - AUTO_DB_INIT=false
      - AUTO_DB_UPDATE=false
    ports:
      - '443:443'
      - '80:80'    # needed only for Let's Encrypt HTTP-01 challenge
    volumes:
      - /var/elabftw/web:/elabftw/uploads
      - /etc/letsencrypt:/ssl
    networks:
      - elabftw-net

  mysql:
    image: mysql:8.4
    restart: always
    container_name: mysql
    healthcheck:
      test: "/usr/bin/mysql --user=$$MYSQL_USER --password=$$MYSQL_PASSWORD
--execute 'SHOW DATABASES;'"
      interval: 5s
      timeout: 5s
      retries: 42
    cap_drop:
      - AUDIT_WRITE
      - MKNOD
      - SYS_CHROOT
      - SETFCAP
      - NET_RAW
    cap_add:
      - SYS_NICE
    environment:
      - MYSQL_ROOT_PASSWORD=CHANGE_THIS_ROOT_PASSWORD
      - MYSQL_DATABASE=elabftw
      - MYSQL_USER=elabftw
      - MYSQL_PASSWORD=CHANGE_THIS_STRONG_PASSWORD
      - TZ=America/Chicago
    volumes:
      - /var/elabftw/mysql:/var/lib/mysql
    expose:
      - '3306'
    networks:
      - elabftw-net
```

7.4 Pull Docker Images

Pull the required Docker images before starting the containers. This ensures images are available and allows you to verify connectivity to Docker Hub:

```
# Change to the configuration directory
cd /etc/elabftw

# Pull all images defined in docker-compose.yml
sudo docker compose pull

# Expected output: Pulling web (elabftw/elabimg:stable)... done
#                  Pulling mysql (mysql:8.4)... done

# Verify images are available locally
sudo docker images
# Expected: elabftw/elabimg  stable  <hash>  <size>
#           mysql             8.4     <hash>  <size>
```

7.5 Start the Containers

```
# Start all containers in detached (background) mode
sudo docker compose -f /etc/elabftw/docker-compose.yml up -d

# Expected output:
# [+] Running 3/3
#  ✔ Network elabftw_elabftw-net    Created
#  ✔ Container mysql                Started
#  ✔ Container elabftw              Started

# Verify both containers are running
sudo docker compose -f /etc/elabftw/docker-compose.yml ps
# Expected: NAME       IMAGE              STATUS
#           elabftw    elabftw/elabimg    Up (healthy)
#           mysql      mysql:8.4          Up (healthy)

# Monitor startup logs in real-time (Ctrl+C to stop following)
sudo docker compose -f /etc/elabftw/docker-compose.yml logs -f
```

7.6 Initialize the Database

After the containers are running, initialize the ELabELN database schema. This creates all tables, indexes, and initial data:

```
# Run the database initialization command inside the elabftw container
sudo docker exec -it elabftw bin/init db:install

# Expected output:
# [OK] Database successfully initialized.

# If the database was already initialized (e.g., after a reinstall), use:
# sudo docker exec -it elabftw bin/init db:update

# Alternative: set AUTO_DB_INIT=true in docker-compose.yml
# This automatically initializes the database on first container start
# (useful for automated deployments but should be disabled after first start)
```

7.7 Verify Installation

```
# Check container health status
```

```
sudo docker ps --format 'table {{.Names}}\t{{.Status}}\t{{.Ports}}'

# Check nginx is serving correctly
curl -k https://localhost/
# Expected: HTML content (the ELabELN login page)

# Check that the API endpoint is accessible
curl -k https://localhost/api/v2/info
# Expected: JSON response with ELabELN version info

# Check MySQL is accessible from the web container
sudo docker exec -it elabftw php bin/init system:info
# Expected: System information including database connection status
```

7.8 First Login and SysAdmin Setup

Navigate to your ELabELN URL in a web browser to complete the initial setup:

6. Open https://elab.yourlab.com in a browser. If the TLS certificate is not yet issued (Let's Encrypt), you may see a browser warning. Accept it temporarily for the first login.
7. On the registration page, create your first account. The first account registered is automatically granted SysAdmin privileges.
8. After logging in as SysAdmin, navigate to Admin Panel → SysConfig to:
   - Configure email (SMTP) settings. This is required before sending invitations.
   - Create your first team.
   - Configure the timestamping service (TSA) if different from the default DFN.de.
9. Create your first Team Admin: go to Admin Panel → Users → promote the first researcher to Admin for the team.
10. Invite additional users via Admin Panel → Users → Add User.

8. Post-Installation Configuration

8.1 Configure Automatic Container Restart on Boot

The restart: always setting in docker-compose.yml ensures containers restart automatically if they crash, but they also restart after a system reboot. Verify this is working by testing a reboot:

```
# Create a systemd service to start Docker Compose on boot (belt and suspenders)
sudo tee /etc/systemd/system/elabftw.service > /dev/null << 'EOF'
[Unit]
Description=ELabELN Docker Compose Service
Requires=docker.service
After=docker.service

[Service]
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=/etc/elabftw
ExecStart=/usr/bin/docker compose up -d
ExecStop=/usr/bin/docker compose down
TimeoutStartSec=0

[Install]
WantedBy=multi-user.target
EOF

# Enable the service
sudo systemctl enable elabftw
sudo systemctl daemon-reload

# Test: reboot and verify containers start automatically
# sudo reboot
# After reboot: sudo docker ps
```

8.2 Set Up Automated Backups

ELabELN has two critical data sources that must be backed up independently: the MySQL database and the uploaded files directory. Both must be backed up together to ensure a consistent restore point.

Option A: elabctl Backup (Built-in, uses borgbackup)

```
# Install borgbackup
sudo apt-get install -y borgbackup

# Initialize the backup repository
# Replace /backup with your backup destination (local, NFS, external drive, etc.)
sudo elabctl backup --init /backup

# Run a manual backup to test
sudo elabctl backup run

# Schedule daily backups via cron
sudo tee /etc/cron.d/elabftw-backup > /dev/null << 'EOF'
# ELabELN Daily Backup — runs at 3:00 AM every day
```

```
0 3 * * * root /usr/bin/elabctl backup run >> /var/log/elabftw-backup.log 2>&1
EOF
```

Option B: Manual Database Dump + File Sync

```
# Create a backup script at /usr/local/bin/elab-backup.sh
sudo tee /usr/local/bin/elab-backup.sh > /dev/null << 'BACKUPEOF'
#!/bin/bash
# ELabELN Backup Script
# Variables
BACKUP_DIR=/backup/elabftw
DATE=$(date +%Y-%m-%d)
DB_USER=elabftw
DB_PASSWORD=CHANGE_THIS_STRONG_PASSWORD
DB_NAME=elabftw

# Create backup directory for today
mkdir -p ${BACKUP_DIR}/${DATE}

# 1. Dump MySQL database from inside the container
docker exec mysql mysqldump \
  --user=${DB_USER} \
  --password=${DB_PASSWORD} \
  --single-transaction \
  --routines \
  --triggers \
  ${DB_NAME} | gzip > ${BACKUP_DIR}/${DATE}/elabftw-db.sql.gz

# 2. Sync uploaded files to backup location
rsync -av --delete /var/elabftw/web/ \
  ${BACKUP_DIR}/${DATE}/uploads/

# 3. Copy configuration
cp /etc/elabftw/docker-compose.yml \
  ${BACKUP_DIR}/${DATE}/docker-compose.yml.bak

# 4. Verify backup
echo "Backup completed: ${BACKUP_DIR}/${DATE}"
ls -lh ${BACKUP_DIR}/${DATE}/

# 5. Remove backups older than 30 days
find ${BACKUP_DIR} -maxdepth 1 -type d -mtime +30 -exec rm -rf {} +
BACKUPEOF

# Make the script executable
sudo chmod +x /usr/local/bin/elab-backup.sh

# Add to cron for daily execution at 2:00 AM
echo '0 2 * * * root /usr/local/bin/elab-backup.sh >> /var/log/elab-backup.log
2>&1' | \
  sudo tee /etc/cron.d/elab-backup

# Test the backup manually
sudo /usr/local/bin/elab-backup.sh
```

Important

Store backup copies on a separate physical or network location from the server. A backup stored on the same disk as the application does not protect against disk failure. Use offsite

storage: S3, Azure Blob, Google Cloud Storage, a remote NFS server, or a separate physical backup appliance.

8.3 Configure Log Rotation

```
# Create logrotate configuration for Docker container logs
sudo tee /etc/logrotate.d/docker-elabftw > /dev/null << 'EOF'
/var/lib/docker/containers/*/*-json.log {
    rotate 7
    daily
    compress
    delaycompress
    missingok
    copytruncate
}
EOF

# Optionally configure Docker daemon log rotation globally
sudo tee /etc/docker/daemon.json > /dev/null << 'EOF'
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "50m",
    "max-file": "7"
  }
}
EOF

# Restart Docker to apply logging configuration
sudo systemctl restart docker

# Restart ELabELN containers after Docker restart
cd /etc/elabftw && sudo docker compose up -d
```

9. Updating ELabELN

ELabELN updates are distributed as new Docker image versions on Docker Hub. The update process pulls the new image, recreates containers, and runs any required database migrations. Updates should always be preceded by a full backup.

9.1 Standard Update Procedure

```
# ========================================================
# STEP 1: Create a full backup before updating
# ========================================================
sudo /usr/local/bin/elab-backup.sh
# Verify backup completed successfully before proceeding

# ========================================================
# STEP 2: Check release notes for breaking changes
# Release notes: https://github.com/elabftw/elabftw/releases
# ========================================================

# ========================================================
# STEP 3: Pull the latest stable image
# ========================================================
cd /etc/elabftw
sudo docker compose pull

# ========================================================
# STEP 4: Recreate containers with the new image
# ========================================================
sudo docker compose up -d --remove-orphans

# ========================================================
# STEP 5: Run database migrations (if required)
# ========================================================
sudo docker exec -it elabftw bin/init db:update
# Or set AUTO_DB_UPDATE=true in docker-compose.yml to run automatically

# ========================================================
# STEP 6: Verify the update was successful
# ========================================================
sudo docker exec -it elabftw bin/init system:info
# Check the version number in the output

# View current running versions
sudo docker ps --format 'table {{.Image}}\t{{.Status}}'
```

> **Note**
> If you have pinned a specific version tag in docker-compose.yml (e.g., image: elabftw/elabimg:5.3.4), update the tag to the new version before pulling. LabLynx recommends using the stable tag for automatic minor updates, and pinning for strictly change-controlled environments.

9.2 Pinning a Specific Version

```
# In /etc/elabftw/docker-compose.yml, under 'web' service:
# Change:
image: elabftw/elabimg:stable
# To a specific version:
image: elabftw/elabimg:5.3.11

# After changing the tag, pull and restart as normal:
sudo docker compose pull
sudo docker compose up -d
sudo docker exec -it elabftw bin/init db:update
```

9.3 Rollback Procedure

If an update introduces problems, roll back to the previous version by restoring the backup and running the previous container image:

```
# Stop current containers
cd /etc/elabftw && sudo docker compose down

# Restore database from backup
# Replace /backup/elabftw/2026-06-01/elabftw-db.sql.gz with your backup path
zcat /backup/elabftw/2026-06-01/elabftw-db.sql.gz | \
  sudo docker exec -i mysql mysql \
  --user=elabftw \
  --password=CHANGE_THIS_STRONG_PASSWORD \
  elabftw

# Restore uploaded files from backup
sudo rsync -av --delete /backup/elabftw/2026-06-01/uploads/ /var/elabftw/web/

# Edit docker-compose.yml to use the previous image version
sudo nano /etc/elabftw/docker-compose.yml
# Change image tag to the previous version: image: elabftw/elabimg:5.3.10

# Restart with the previous version
sudo docker compose up -d

# Verify rollback was successful
sudo docker exec -it elabftw bin/init system:info
```

10. Security Hardening

The following hardening steps are strongly recommended for all production deployments, particularly in regulated environments (pharma, clinical, forensic) where data integrity and access control are compliance requirements.

10.1 SSH Security

```
# Edit SSH server configuration
sudo nano /etc/ssh/sshd_config

# Recommended settings:
PermitRootLogin no          # Disable direct root SSH login
PasswordAuthentication no   # Require SSH key authentication
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
X11Forwarding no
MaxAuthTries 3

# Restart SSH (you must have an SSH key loaded before doing this)
sudo systemctl restart sshd

# Verify SSH still works in a separate terminal before closing current session
```

10.2 Install fail2ban

```
# Install fail2ban for SSH and web brute-force protection
sudo apt-get install -y fail2ban

# Create a local configuration
sudo tee /etc/fail2ban/jail.local > /dev/null << 'EOF'
[DEFAULT]
bantime  = 3600
findtime = 600
maxretry = 5
backend = auto

[sshd]
enabled = true
port = 22
logpath = /var/log/auth.log
maxretry = 3

[nginx-http-auth]
enabled = true
filter = nginx-http-auth
port = http,https
logpath = /var/log/nginx/error.log
EOF

sudo systemctl enable fail2ban
sudo systemctl start fail2ban
sudo fail2ban-client status
```

10.3 Docker and ufw Integration

Docker bypasses ufw rules by default, which means Docker-exposed ports are accessible even if ufw would block them. Fix this by restricting Docker port bindings:

```
# Option A: Restrict Docker port binding to localhost only
# (Use this if a reverse proxy handles external access)
# In docker-compose.yml, change the ports section from:
    ports:
      - '443:443'
# To:
    ports:
      - '127.0.0.1:443:443'

# Option B: Install ufw-docker (fixes Docker iptables integration with ufw)
# Download and install ufw-docker
sudo wget -O /usr/local/bin/ufw-docker \
  https://github.com/chaifeng/ufw-docker/raw/master/ufw-docker
sudo chmod +x /usr/local/bin/ufw-docker
sudo ufw-docker install

# After installing ufw-docker, allow specific container ports
sudo ufw-docker allow elabftw 443/tcp
sudo ufw-docker allow elabftw 80/tcp

# Reload ufw
sudo ufw reload
```

10.4 File System Security

```
# Set appropriate permissions on ELabELN data directories
# The uploads directory: readable/writable by nginx user (UID 101) only
sudo chown -R 101:101 /var/elabftw/web
sudo chmod -R 750 /var/elabftw/web

# The MySQL data directory: readable by the Docker mysql container only
sudo chown -R 999:999 /var/elabftw/mysql
sudo chmod -R 750 /var/elabftw/mysql

# Protect the docker-compose.yml (contains passwords)
sudo chown root:root /etc/elabftw/docker-compose.yml
sudo chmod 600 /etc/elabftw/docker-compose.yml

# Protect the backup script (contains database password)
sudo chown root:root /usr/local/bin/elab-backup.sh
sudo chmod 700 /usr/local/bin/elab-backup.sh
```

10.5 Enable Automatic Security Updates

```
# Install unattended-upgrades for automatic security patching
sudo apt-get install -y unattended-upgrades

# Enable security updates
sudo dpkg-reconfigure -plow unattended-upgrades
# Select 'Yes' when prompted

# Configure to automatically reboot if needed
sudo tee -a /etc/apt/apt.conf.d/50unattended-upgrades > /dev/null << 'EOF'
Unattended-Upgrade::Automatic-Reboot "true";
Unattended-Upgrade::Automatic-Reboot-Time "03:00";
```

```
EOF

# Note: ELabELN containers will restart automatically after reboot
# due to the restart: always policy in docker-compose.yml
```

10.6 TLS Security Configuration

The nginx configuration inside the ELabELN container is pre-configured with modern TLS settings. For reference, the key TLS parameters in the nginx configuration are:

Parameter	Configuration
TLS Protocols	TLSv1.2, TLSv1.3 only. TLS 1.0 and 1.1 are disabled.
Cipher Suites	Modern cipher list prioritizing AEAD ciphers (AES-GCM, ChaCha20-Poly1305). Prioritizes forward secrecy.
HSTS	HTTP Strict Transport Security header is set. After first visit, browsers enforce HTTPS for the domain.
OCSP Stapling	Certificate revocation status is checked and cached by nginx to speed up TLS handshake.
Session Cache	TLS session resumption cache enabled for performance.
Security Headers	X-Frame-Options: SAMEORIGIN, X-Content-Type-Options: nosniff, X-XSS-Protection, Content-Security-Policy are set.

11. Monitoring and Health Checks

11.1 Container Health Monitoring

```
# Check container status and health
sudo docker ps

# Detailed health check status
sudo docker inspect --format '{{.Name}} {{.State.Health.Status}}' $(sudo docker ps
-q)

# View recent logs from the web container
sudo docker logs elabftw --tail 50

# View real-time logs (Ctrl+C to stop)
sudo docker logs -f elabftw

# View MySQL logs
sudo docker logs mysql --tail 50

# Monitor container resource usage
sudo docker stats --no-stream
# Output: CPU%, MEM USAGE, NET I/O, BLOCK I/O for each container
```

11.2 PHP-FPM Status Endpoint

The ELabELN container exposes a /php-status endpoint for PHP-FPM pool metrics. Set STATUS_PASSWORD in docker-compose.yml to enable it:

```
# In docker-compose.yml, set STATUS_PASSWORD under the web service:
- STATUS_PASSWORD=your-monitor-password

# After restarting the container, query the status endpoint
# (use the 'elabftw' username and your STATUS_PASSWORD)
curl -u elabftw:your-monitor-password \
     https://elab.yourlab.com/php-status

# Example output:
# pool:                 www
# process manager:      dynamic
# start time:           01/Jun/2026:09:00:00 +0000
# start since:          86400
# accepted conn:        12450
# listen queue:         0
# max listen queue:     0
# listen queue len:     0
# idle processes:       8
# active processes:     2
# total processes:      10
# max active processes: 15
# max children reached: 0
# slow requests:        3
```

11.3 Database Monitoring

```
# Connect to MySQL for manual inspection
sudo docker exec -it mysql mysql \
  --user=elabftw \
  --password=CHANGE_THIS_STRONG_PASSWORD \
  elabftw

# Inside MySQL prompt:
# SHOW TABLE STATUS;              -- View all tables and their sizes
# SHOW PROCESSLIST;               -- View current queries
# SHOW STATUS LIKE 'connections'; -- View connection count
# SHOW ENGINE INNODB STATUS;      -- InnoDB engine status

# Check database size
sudo docker exec -it mysql mysql \
  --user=elabftw \
  --password=CHANGE_THIS_STRONG_PASSWORD \
  --execute="SELECT table_schema AS 'Database', \
  ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) AS 'Size (MB)' \
  FROM information_schema.tables \
  GROUP BY table_schema;"
```

11.4 Disk Space Monitoring

```
# Check overall disk usage
df -h /var/elabftw

# Check size of uploads directory
du -sh /var/elabftw/web

# Check size of MySQL data directory
du -sh /var/elabftw/mysql

# Set up a disk usage alert (cron job sends email when disk > 80%)
sudo tee /usr/local/bin/check-disk.sh > /dev/null << 'EOF'
#!/bin/bash
THRESHOLD=80
USAGE=$(df /var/elabftw | awk 'NR==2 {print $5}' | tr -d '%')
if [ "$USAGE" -gt "$THRESHOLD" ]; then
  echo "WARNING: /var/elabftw disk usage is ${USAGE}%" | \
    mail -s "ELabELN Disk Alert" admin@yourlab.com
fi
EOF
sudo chmod +x /usr/local/bin/check-disk.sh
echo '0 * * * * root /usr/local/bin/check-disk.sh' | sudo tee
/etc/cron.d/disk-check
```

12. Troubleshooting

Issue	Symptom	Resolution
Containers fail to start	docker compose logs shows connection error	Verify DB_PASSWORD matches MYSQL_PASSWORD. Wait 30 seconds for MySQL healthcheck to pass. Check docker compose ps for healthy status on mysql container.
Cannot access HTTPS (connection refused)	curl: (7) Failed to connect	Verify port 443 is open: sudo ufw status. Check Docker port binding: sudo docker ps shows 0.0.0.0:443->443/tcp. Verify nginx started: sudo docker logs elabftw | grep nginx.
Certificate error in browser	SSL_ERROR_RX_RECORD_TOO_LONG	DISABLE_HTTPS=true but accessing via HTTPS. If using a reverse proxy, ensure it handles TLS and forwards plain HTTP to the container on a different port.
Let's Encrypt certificate fails	certbot log shows challenge failed	Port 80 must be open and accessible from the internet for HTTP-01 validation. Verify SERVER_NAME exactly matches your DNS FQDN. Check that no firewall is blocking port 80.
500 Internal Server Error	Browser shows 500, nginx logs show PHP-FPM error	Check PHP-FPM logs: sudo docker exec elabftw cat /var/log/php-fpm.log. Common causes: insufficient PHP memory (increase MAX_PHP_MEMORY), failed database connection, missing file permissions on /var/elabftw/web.
Database initialization fails	bin/init db:install returns error	Verify MySQL is running and healthy: sudo docker ps. Check database credentials in docker-compose.yml. Try manual connection: sudo docker exec -it mysql mysql -u elabftw -p elabftw.
Uploads directory permission denied	PHP writes fail, attachments don't save	Run: sudo chown -R 101:101 /var/elabftw/web. The nginx user (UID 101) must own the uploads directory.
Container keeps restarting	docker ps shows restarting status	Check logs immediately after start: sudo docker logs elabftw --tail 20. Common cause: environment variable misconfiguration or missing SECRET_KEY.
High memory usage	docker stats shows >90% memory	Reduce PHP_MAX_CHILDREN or MAX_PHP_MEMORY. Add physical RAM. Check for memory leaks: sudo docker stats shows consistent growth over hours.
MySQL taking too long to start	web container shows 'waiting for mysql'	The mysql healthcheck retries 42 times with 5s intervals (3.5 minutes total). If MySQL doesn't start in that time, check MySQL logs: sudo docker logs mysql. Ensure

		/var/elabftw/mysql has correct permissions and sufficient disk space.

12.1 Useful Diagnostic Commands

```
# View all container logs with timestamps
sudo docker compose -f /etc/elabftw/docker-compose.yml logs --timestamps

# Enter the web container shell for direct debugging
sudo docker exec -it elabftw /bin/sh

# Enter the MySQL container
sudo docker exec -it mysql /bin/bash

# Check nginx configuration for syntax errors
sudo docker exec -it elabftw nginx -t

# Check PHP configuration
sudo docker exec -it elabftw php --ini

# View ELabELN system info
sudo docker exec -it elabftw bin/init system:info

# Clear application cache
sudo docker exec -it elabftw bin/init tools:clearcache

# Check for available database updates
sudo docker exec -it elabftw bin/init db:updatecheck

# Restart only the nginx and PHP services inside the container (no downtime)
sudo docker exec -it elabftw s6-svc -r /run/service/nginx
sudo docker exec -it elabftw s6-svc -r /run/service/php-fpm
```

ELabELN — License, Technical Specifications & Installation Manual

Version 1.0 | June 2026

LabLynx, Inc. | www.lablynx.com | support@lablynx.com | www.elabeln.com

ELabELN is powered by LabLynx and built on eLabFTW, an open-source ELN (AGPL-3.0). Source: https://github.com/elabftw/elabftw

© 2026 LabLynx, Inc. All rights reserved. This document is CONFIDENTIAL.